

日 本 国 特 許 庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日 2 0 0 3 年 3 月 2 8 日
Date of Application:

出 願 番 号 特 願 2 0 0 3 - 0 9 2 3 7 1
Application Number:
[ST. 10/C] : [J P 2 0 0 3 - 0 9 2 3 7 1]

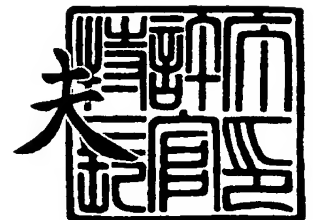
出 願 人 セイコーエプソン株式会社
Applicant(s):



2 0 0 4 年 1 月 2 6 日

特許庁長官
Commissioner,
Japan Patent Office

今 井 康 夫



出証番号 出証特 2 0 0 4 - 3 0 0 2 7 0 0

【書類名】 特許願

【整理番号】 J0097495

【提出日】 平成15年 3月28日

【あて先】 特許庁長官殿

【国際特許分類】 H04L 12/00

【発明者】

 【住所又は居所】 長野県諏訪市大和3丁目3番5号 セイコーエプソン株式会社内

 【氏名】 磯村 政一

【特許出願人】

 【識別番号】 000002369

 【氏名又は名称】 セイコーエプソン株式会社

【代理人】

 【識別番号】 100066980

 【弁理士】

 【氏名又は名称】 森 哲也

【選任した代理人】

 【識別番号】 100075579

 【弁理士】

 【氏名又は名称】 内藤 嘉昭

【選任した代理人】

 【識別番号】 100103850

 【弁理士】

 【氏名又は名称】 崔 秀▲てつ▼

【手数料の表示】

 【予納台帳番号】 001638

 【納付金額】 21,000円

【提出物件の目録】

 【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 0014966

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 ベクトルプロセッサおよびレジスタのアドレス指定方法

【特許請求の範囲】

【請求項 1】 複数の要素データからなるベクトルデータを、レジスタを使用して演算処理するベクトルプロセッサであって、

複数の要素レジスタからなるベクトルレジスタとして使用可能なレジスタと、
該ベクトルレジスタにおける任意の要素レジスタのアドレスを先頭として、前記ベクトルレジスタのアドレスを、循環的に指定するアドレス指定回路と、
を備えることを特徴とするベクトルプロセッサ。

【請求項 2】 前記レジスタは、複数のスカラーレジスタが 1 組となり、該 1 組のスカラーレジスタのいずれかのアドレスが先頭として指定されることにより、該複数のスカラーレジスタそれぞれのアドレスが循環的に指定されることを特徴とする請求項 1 記載のベクトルプロセッサ。

【請求項 3】 前記レジスタは、任意の要素レジスタを先頭として指定可能なベクトルレジスタによって構成されていることを特徴とする請求項 1 記載のベクトルプロセッサ。

【請求項 4】 前記レジスタに記憶されたデータを対象としてベクトル演算を行う際、前記ベクトルレジスタにおいて先頭として指定されたアドレスから、該ベクトルレジスタの要素データを順次読み出し、該ベクトルレジスタの末尾のアドレスに達した場合、先頭のアドレスに戻って要素データの読み出しを継続可能であることを特徴とする請求項 1～3 のいずれかに記載のベクトルプロセッサ。

【請求項 5】 ベクトル演算の演算結果を前記レジスタに書き込む際、前記ベクトルレジスタにおいて先頭として指定されたアドレスから、該ベクトルレジスタの要素データを順次書き込み、該ベクトルレジスタの末尾のアドレスに達した場合、先頭のアドレスに戻って要素データの書き込みを継続可能であることを特徴とする請求項 1～4 のいずれかに記載のベクトルプロセッサ。

【請求項 6】 複数の要素データからなるベクトルデータの演算処理に用いるレジスタのアドレス指定方法であって、

所定の要素レジスタを複数の要素レジスタからなるベクトルレジスタとして取り扱い、該ベクトルレジスタにおける任意の要素レジスタのアドレスを先頭として指定すると、前記ベクトルレジスタにおける各要素レジスタのアドレスが循環的に指定されることを特徴とするレジスタのアドレス指定方法。

【発明の詳細な説明】

【0 0 0 1】

【発明の属する技術分野】

本発明は、ベクトルレジスタを使用して演算を行うベクトルプロセッサおよびレジスタのアドレス指定方法に関する。

【0 0 0 2】

【従来の技術】

従来、科学技術計算における繰り返し演算や、画像処理における画素データの演算あるいは配列データの演算等において、ベクトル演算が行われている。

ベクトル演算においては、メモリからベクトルデータを読み出し、ベクトルレジスタに記憶した上で、ベクトルデータ同士の加算あるいは乗算等、所定のベクトル演算が行われる。

【0 0 0 3】

例えば、配列において隣接する 2 つのデータを対象として、加算等の演算を行う場合、1 つのベクトルレジスタに配列データを先頭から n 個 (n は自然数) 分記憶し、他のベクトルレジスタには、配列データを 2 番目のものから n 個分記憶する。そして、それら 2 つのベクトルレジスタを対象として加算等の演算命令を実行することにより、ベクトルレジスタ内における同一アドレス同士の各要素データが演算され、配列データの演算が一括して行われる。

【0 0 0 4】

ここで、ベクトル演算に関連する技術として、特開昭 6 0 - 2 4 6 7 2 号公報に記載された技術が知られている。

本号公報においては、上述のように、配列において隣接する 2 つのデータを対象として演算を行う際、演算処理における効率を向上させる技術が開示されている。

【0 0 0 5】

即ち、上述の演算を行う場合、配列の 2 番目から $n - 1$ 個（演算される配列データの末尾の 1 つ前）のデータは、2 つのレジスタに記憶するために、それぞれ 2 度ずつ読み出されることとなる。

そこで、本号公報に記載された技術では、演算対象となる配列を先頭から $n + 1$ 個（演算される配列データの末尾）まで一度だけ読み出し、所定のレジスタに記憶する。そして、そのレジスタ内のデータを複数のベクトルレジスタに分配して記憶させる分配回路によって、先頭から所定数分の配列データを格納するベクトルレジスタと、2 番目のものから所定数分格納するベクトルレジスタとに振り分けて記憶する。

【0 0 0 6】

このような手順とすることにより、メモリから配列データを読み出す際に、同一のデータが重複して読み出されることを回避し、効率的にベクトル演算を行おうとするものである。

【0 0 0 7】

【特許文献 1】

特開昭 6 0 - 2 4 6 7 2 号公報

【0 0 0 8】

【発明が解決しようとする課題】

しかしながら、特開昭 6 0 - 2 4 6 7 2 号公報に記載された技術においては、メモリから読み出したベクトルデータを一旦、所定のレジスタに記憶した後に、2 つのベクトルレジスタに分配するため、多くのレジスタリソースが必要となる。

【0 0 0 9】

したがって、回路規模が大きくなるという問題や、レジスタリソースが圧迫され、処理効率が低下するという問題が生じていた。

さらに、近年、機器の低消費電力化が重要視されており、書き込みを行う場合にのみ、レジスタを構成するフリップフロップに電力を供給することが一般的である。一方、特開昭 6 0 - 2 4 6 7 3 号公報に記載された技術においては、同一

のデータを同時に複数のベクトルレジスタに記憶することとなるため、同時にクロックを供給する対象が増え、消費電力が増大するという問題があった。

【0 0 1 0】

本発明の課題は、ベクトルレジスタを用いたベクトル演算を効率的に行うことである。

【0 0 1 1】

【課題を解決するための手段】

以上の課題を解決するため、本発明は、

複数の要素データからなるベクトルデータを、レジスタを使用して演算処理するベクトルプロセッサであって、複数の要素レジスタからなるベクトルレジスタとして使用可能なレジスタ（例えば、図 6 のレジスタファイル 4 0）と、該ベクトルレジスタにおける任意の要素レジスタのアドレスを先頭として、前記ベクトルレジスタのアドレスを、循環的に指定するアドレス指定回路（例えば、図 1 0 に示す第 1 ソースレジスタ決定回路 7 2 等）とを備えることを特徴としている。

【0 0 1 2】

また、前記レジスタは、複数のスカラーレジスタが 1 組となり、該 1 組のスカラーレジスタのいずれかのアドレスが先頭として指定されることにより、該複数のスカラーレジスタそれぞれのアドレスが循環的に指定されることを特徴としている。

また、前記レジスタは、任意の要素レジスタを先頭として指定可能なベクトルレジスタによって構成されていることを特徴としている。

【0 0 1 3】

また、前記レジスタに記憶されたデータを対象としてベクトル演算を行う際、前記ベクトルレジスタにおいて先頭として指定されたアドレスから、該ベクトルレジスタの要素データを順次読み出し、該ベクトルレジスタの末尾のアドレスに達した場合、先頭のアドレスに戻って要素データの読み出しを継続可能であることを特徴としている。

【0 0 1 4】

また、ベクトル演算の演算結果を前記レジスタに書き込む際、前記ベクトルレ

レジスタにおいて先頭として指定されたアドレスから、該ベクトルレジスタの要素データを順次書き込み、該ベクトルレジスタの末尾のアドレスに達した場合、先頭のアドレスに戻って要素データの書き込みを継続可能であることを特徴としている。

【0015】

また、本発明は、

複数の要素データからなるベクトルデータの演算処理に用いるレジスタのアドレス指定方法であって、所定の要素レジスタを複数の要素レジスタからなるベクトルレジスタとして取り扱い、該ベクトルレジスタにおける任意の要素レジスタのアドレスを先頭として指定すると、前記ベクトルレジスタにおける各要素レジスタのアドレスが循環的に指定されることを特徴としている。

【0016】

本発明によれば、リングバッファをなすベクトルレジスタとして使用可能なレジスタを備え、そのリングバッファにおける任意のアドレスを先頭アドレスとして指定可能である。

そのため、演算対象である複数のベクトルデータが重複する場合に、ベクトルデータそれぞれを異なるベクトルレジスタに記憶することなく、1つのベクトルレジスタに記憶されたベクトルデータを循環的に読み込みあるいは書き込むことができる。

【0017】

したがって、同一のデータが重複して読み出されることを回避できると共に、必要となるレジスタリソースを減少させることが可能となり、ベクトルレジスタを用いたベクトル演算を効率的に行うことが可能となる。

【0018】

【発明の実施の形態】

以下、図を参照して本発明に係るベクトルプロセッサの実施の形態を説明する。

本発明に係るベクトルプロセッサは、リングバッファをなすベクトルレジスタを備え、ベクトルレジスタの任意のアドレスからデータにアクセスすることを可

能としている。

【0019】

したがって、初めに、本発明の基本となる考え方について説明する。なお、ここでは、配列において隣接する2つのデータを対象として加算を行う場合（例えば、画像処理において、隣接画素同士の平均値を求める処理）を例に挙げて説明する。

図1は、8つの要素レジスタR0～R7を有するベクトルレジスタVRを示す図である。

【0020】

本発明においては、加算対象となるベクトルデータを記憶するベクトルレジスタを2つ用いることなく、1つのベクトルレジスタVRによって、2つのベクトルレジスタを用いる場合と同様の処理を行う。なお、ここでいうベクトルレジスタには、スカラーレジスタをベクトルレジスタとして使用する場合を含むものである。

【0021】

まず、ベクトルレジスタVRを備えるベクトルプロセッサに、8つの要素データx0～x7からなるベクトルデータのロード命令と、そのロード命令から2サイクル遅れた加算命令とを与える。

すると、サイクル“0”において、ベクトルレジスタVRの要素レジスタR0に要素データx0が書き込まれ、1サイクル毎に、引き続く要素レジスタに後続の要素データが順次書き込まれていく。

【0022】

ここで、サイクル“2”において、サイクル“0”に開始されたロード命令から2サイクル遅れて、加算命令が実行される。

図2は、サイクル“2”におけるベクトルレジスタVRの状態を示す図である。

図2において、要素レジスタR0、R1には、それぞれ要素データx0、x1が既に記憶されており、さらに要素レジスタR2には、要素データx2が書き込まれている。また、図2においては、加算命令が開始されており、要素レジスタ

R 0, R 1 に記憶された要素データが加算されている。

【0023】

このように、ロード命令から2サイクル遅れて加算命令を実行していくと、サイクル“7”においては、図3に示す状態となる。

図3において、ロード命令については、要素レジスタ R 7 に要素データ x 7 を書き込む状態であり、サイクル“0”で開始されたロード命令はサイクル“7”で終了する。なお、加算命令については、要素レジスタ R 5, R 6 に記憶された要素データが加算されている。

【0024】

次に、ベクトルプロセッサには、後続のデータを処理するため、サイクル“8”において、2つ目のロード命令が与えられる。

すると、サイクル“8”においては、ベクトルレジスタ V R は、図4に示す状態となる。

図4において、ロード命令については、先頭に戻り、要素レジスタ R 0 に要素データ x 8 を書き込む状態であり、加算命令については、要素レジスタ R 6, R 7 に記憶された要素データが加算されている。なお、サイクル“2”で開始された加算命令は、ロード命令から2サイクル送れているため、依然として加算処理を実行している。

【0025】

続いて、サイクル“2”で開始された加算命令の8サイクル目（最終サイクル）であるサイクル“9”に移行する。

図5は、サイクル“9”におけるベクトルレジスタ V R の状態を示す図である。

図5において、加算命令については、要素レジスタ R 7 から、加算対象の1つとなる要素データ x 7 が読み出されている。

【0026】

ここで、加算対象となる他の要素データについては、ベクトルレジスタ V R の先頭に戻り、要素レジスタ R 0 に記憶されている要素データ x 8 が読み出されている。即ち、要素レジスタ R 7, R 0 に記憶された要素データ x 7, x 8 が加算

される。

なお、この後、サイクル“0”～“9”が適宜繰り返される。

【0027】

このように、加算命令によって、ベクトルレジスタの最終アドレスを超えた要素データが参照される場合であっても、ベクトルレジスタがリングバッファをなすことによって、参照される要素データを容易に読み出すことができ、引き続き処理を円滑に行うことができる。

また、同一の要素データをメモリ等から重複して読み出す必要がなく、また、8要素を記憶可能なベクトルレジスタを1つ用いることで、8要素を超えるベクトルデータの要素データ同士の演算を行うことができるため、ベクトル演算を効率的に行うことが可能となる。

【0028】

次に、本発明に係るベクトルプロセッサの構成を説明する。

図6は、本発明を適用したベクトルプロセッサ1の構成を示す図である。

図6において、ベクトルプロセッサ1は、メモリ10と、メモリ制御部20と、命令フェッチ部30と、レジスタファイル40と、ロードユニット50と、ストアユニット60と、演算ユニット70とを含んで構成される。

【0029】

メモリ10は、ベクトルプロセッサ1に与えられる命令コードおよび演算対象となるデータを記憶している。

図7は、命令コードのデータ形式の一例を示す図であり、(a)は、ロード命令の形式、(b)は、ストア命令の形式、(c)は、演算命令の形式を示している。図7において、各命令コードには、命令コードの命令の種類を示すオペレーションコード、命令の処理対象となるベクトルデータの要素数あるいはレジスタの指定コード等、命令を実行するために必要となる情報が含まれている。

【0030】

メモリ制御部20は、メモリ10に対するアクセス、即ち、データの読み出しや書き込みを制御する。例えば、メモリ制御部20は、ロードユニット50あるいはストアユニット60によって指定されたメモリ10のアドレスからデータを

読み出したり、メモリ 1 0 から読み出されたデータをレジスタファイル 4 0 に出力したりする。

【0 0 3 1】

命令フェッチ部 3 0 は、メモリ制御部 2 0 を介して、メモリ 1 0 から命令コードをフェッチし、一時的に記憶する。

レジスタファイル 4 0 は、メモリ 1 0 から読み出されたデータおよび演算結果を一時的に記憶する。

ロードユニット 5 0 は、命令フェッチ部 3 0 に記憶された命令コードがロード命令である場合に、メモリ 1 0 から命令コードあるいはデータを読み出す処理を行う。

【0 0 3 2】

ストアユニット 6 0 は、命令フェッチ部 3 0 に記憶された命令コードがストア命令である場合に、メモリ 1 0 にデータを書き込む処理を行う。

演算ユニット 7 0 は、命令フェッチ部 3 0 に記憶された命令コードが所定の演算命令である場合に、レジスタファイル 4 0 に記憶された所定データを対象として演算処理を行う。

【0 0 3 3】

ここで、レジスタファイル 4 0 および演算ユニット 7 0 について、詳細に説明する。

まず、レジスタファイル 4 0 について説明する。

レジスタファイル 4 0 は、図 6 に示すように、読み書き可能な 3 2 個のレジスタ R 0 ~ R 3 1 を含んで構成される。

【0 0 3 4】

また、レジスタファイル 4 0 において、レジスタ R 0 ~ R 7、レジスタ R 8 ~ R 1 5、レジスタ R 1 6 ~ R 2 3 およびレジスタ R 2 4 ~ R 3 1 は、それぞれを 1 組として、リングバッファの機能を有するベクトルレジスタとして使用することが可能である。

ここで、リングバッファの機能を有するベクトルレジスタとして、レジスタ R 0 ~ R 3 1 を使用可能とするためには、ベクトルレジスタにリングバッファの機

能を備え、任意のアドレスを先頭アドレスとして指定可能とするものの他、スカラーレジスタをベクトルレジスタとして用いることが可能である。

【0035】

即ち、所定のスカラーレジスタを1組とし、組単位でレジスタを指定可能とすることにより、スカラーレジスタをベクトルレジスタとして使用することが可能である。なお、この場合、1組のスカラーレジスタにおける任意のアドレスを先頭アドレスとして指定することができると共に、スカラーレジスタは本来、レジスタ毎の指定が可能であることから、リングバッファとして、循環的にアドレスを指定することができる。

【0036】

また、図6に示すレジスタファイル40において、レジスタR0～R31には、それぞれに5ビットのコードが割り当てられている。

図8は、レジスタR0～R31それぞれに割り当てられたコードを示す図である。

レジスタファイル40に対し、図8に示す所定のコードを選択信号として入力することにより、対応するレジスタの読み出しおよび書き込みを行うことができる。

【0037】

なお、図8において、5ビットのコードのうち、上位2ビットはベクトルレジスタを指定するコードであり、下位3ビットは、ベクトルレジスタ内のアドレスを指定するコードである。

続いて、演算ユニット70について説明する。

図9は、演算ユニット70の内部構成を示すブロック図である。

【0038】

図9において、演算ユニット70は、命令パイプライン制御部71と、第1ソースレジスタ決定回路72と、第2ソースレジスタ決定回路73と、デスティネーションレジスタ決定回路74と、演算器75と、パイプラインレジスタ（PR）76～79とを含んで構成される。

命令パイプライン制御部71は、演算ユニット70全体を制御するものである

。

【0039】

第1ソースレジスタ決定回路72は、命令コードに含まれる第1ソースレジスタ指定コードに基づいて、第1ソースレジスタを選択する信号（第1ソースレジスタ選択信号）を生成する。

第2ソースレジスタ決定回路73は、命令コードに含まれる第2ソースレジスタ指定コードに基づいて、第2ソースレジスタを選択する信号（第2ソースレジスタ選択信号）を生成する。

【0040】

デスティネーションレジスタ決定回路74は、命令コードに含まれるデスティネーションレジスタ指定コードに基づいて、デスティネーションレジスタを選択する信号（デスティネーションレジスタ選択信号）を生成する。

ここで、第1ソースレジスタ決定回路72、第2ソースレジスタ決定回路73およびデスティネーションレジスタ決定回路74の構成について説明する。

【0041】

なお、第1ソースレジスタ決定回路72、第2ソースレジスタ決定回路73およびデスティネーションレジスタ決定回路74の構成は同様であるため、第1ソースレジスタを例に挙げて説明する。

図10は、第1ソースレジスタ決定回路72の構成例を示す図である。

図10において、第1ソースレジスタ決定回路72は、制御部72aと、セレクト72bと、インクリメンタ72cと、カウンタ72dと、レジスタ72eとを含んで構成される。

【0042】

制御部72aは、命令パイプライン制御部71によって入力される動作開始信号と、命令フェッチ部30によって入力されるベクトル要素数に基づいて、第1ソースレジスタ決定回路72全体を制御する。

セレクト72bは、サイクル“0”においては、命令フェッチ部30から入力された第1ソースレジスタ指定コードを選択して出力し、サイクル“0”以外においては、カウンタ72dおよびレジスタ72eから入力された第1ソースレジ

スタ選択信号を選択して出力する。

【0043】

インクリメンタ72cは、5ビットの第1ソースレジスタ指定コードのうち、下位3ビットを受け取り、“1”加算してカウンタ72dに出力する。

カウンタ72dは、サイクル“0”において、インクリメンタ72cから入力された3ビットのコードを記憶する。

また、カウンタ72dは、制御部72aの指示によってカウントイネーブル状態が切り替えられ、カウントイネーブル状態である場合に、記憶しているコードを+1して更新するカウントアップ動作を行う。

【0044】

レジスタ72eは、第1ソースレジスタ指定コードのうち、上位2ビットを受け取り、1つのベクトル演算が行われている間、そのコードを保持する。

このような構成により、第1ソースレジスタ決定回路72においては、例えば、第1ソースレジスタ指定コードがレジスタR18を示すコード“10010”であり、ベクトル要素数が“8”である場合、第1ソースレジスタ選択信号として、順次、“10010”、“10011”、“10100”、“10101”、“10110”、“10111”が出力され、引き続き、“10000”、“10001”が出力される。即ち、第1ソースレジスタ選択信号によって、レジスタR18→レジスタR19→レジスタR20→レジスタR21→レジスタR22→レジスタR23→レジスタR16→レジスタR17の順に選択される。

【0045】

つまり、レジスタR16～R23を、リングバッファをなすベクトルレジスタとして使用できると共に、これらのレジスタにおける任意のアドレスを先頭アドレスとして指定することが可能となる。

図9に戻り、演算器75は、命令パイプライン制御部71の指示に基づいて、実際に加算等の演算を行う。

【0046】

PR76～79は、パイプライン処理の各ステージにおいて処理されるデータを記憶する。

次に、動作を説明する。

初めに、図 6 を参照して、ベクトルプロセッサ 1 全体の動作について説明する。

【0047】

ベクトルプロセッサ 1 において処理が行われる場合、メモリ制御部 20 を介してメモリ 10 から命令フェッチ部 30 に命令コードが読み出される。

そして、ロードユニット 50、ストアユニット 60 および演算ユニット 70 のそれぞれに、命令フェッチ部 30 から命令コードが出力される。

命令コードが入力されたロードユニット 50、ストアユニット 60 および演算ユニット 70 は、その命令コードをデコードし、それぞれのユニットに対応する命令である場合にのみ、命令を実行する。

【0048】

以下、命令コードの内容に分けて、動作を説明する。

(命令コードがロード命令である場合)

命令フェッチ部 30 から入力された命令コードがロード命令である場合、ロードユニット 50 は、命令コード (図 7 (a) 参照) において指定された基底アドレスレジスタおよびアドレス修飾レジスタのそれぞれを選択する信号をレジスタファイル 40 に出力する。

【0049】

すると、それぞれのアドレスに記憶されている値 (基底アドレス値およびアドレス修飾値) が、ロードユニット 50 に読み込まれる。

そして、ロードユニット 50 は、基底アドレス値およびアドレス修飾値に基づいて、メモリ 10 のロードアドレス (読み込み対象アドレス) を生成し、メモリ制御部 20 に出力する。

【0050】

ロードアドレスが入力されると、メモリ制御部 20 は、メモリ 10 における対応するアドレスのデータ (ロードデータ) を読み出し、そのロードデータをレジスタファイル 40 に出力する。このとき、ロードユニット 50 は、メモリ制御部 20 からレジスタファイル 40 にロードデータが出力されるタイミングに合わせ

て、命令コードにおいて指定されているデスティネーションレジスタを選択するための信号を、レジスタファイル 4 0 に出力する。

【0 0 5 1】

すると、レジスタファイル 4 0 において、デスティネーションレジスタに、ロードデータが書き込まれる。

(命令コードがストア命令である場合)

命令フェッチ部 3 0 から入力された命令コードがストア命令である場合、ストアユニット 6 0 は、命令コード (図 7 (b) 参照) において指定されたデスティネーションレジスタ選択信号をレジスタファイル 4 0 に出力する。

【0 0 5 2】

すると、デスティネーションアドレスに記憶されている値 (ストアデータ) がストアユニット 6 0 に読み込まれる。

そして、ストアユニット 6 0 は、読み込んだストアデータをメモリ制御部 2 0 に出力する。

また、ストアユニット 6 0 は、命令コードにおいて指定されている基底アドレスレジスタおよびアドレス修飾レジスタのそれぞれを選択する信号をレジスタファイル 4 0 に出力する。

【0 0 5 3】

すると、それぞれのアドレスに記憶されている基底アドレス値およびアドレス修飾値が、ストアユニット 6 0 に読み込まれる。

そして、ストアユニット 6 0 は、基底アドレス値およびアドレス修飾値に基づいて、メモリ 1 0 のストアアドレス (書き込み対象アドレス) を生成し、ストアデータをメモリ制御部 2 0 に出力するタイミングと合わせて、ストアアドレスをメモリ制御部 2 0 に出力する。

【0 0 5 4】

ストアデータおよびストアアドレスが入力されると、メモリ制御部 2 0 は、メモリ 1 0 における対応するアドレスに、ストアデータを書き込む。

(命令コードが演算命令である場合)

命令フェッチ部 3 0 から入力された命令コードが演算命令である場合、演算ユ

ニット 70 は、命令コード（図 7（c）参照）において指定された第 1 ソースレジスタ選択信号および第 2 ソースレジスタ選択信号をレジスタファイル 40 に出力する。

【0055】

すると、それぞれのアドレスに記憶されている値（第 1 ソースデータおよび第 2 ソースデータ）が、演算ユニット 70 に読み込まれる。

そして、演算ユニット 70 は、第 1 ソースデータと第 2 ソースデータとの演算を行い、演算結果をレジスタファイル 40 に出力する。このとき、演算ユニット 70 は、演算結果がレジスタファイル 40 に出力されるタイミングに合わせて、命令コードにおいて指定されているデスティネーションレジスタ選択信号を、レジスタファイル 40 に出力する。

【0056】

すると、レジスタファイル 40 において、デスティネーションレジスタに、演算結果が書き込まれる。

次に、演算ユニット 70 の動作について、図 9 を参照して詳細に説明する。

演算ユニット 70 には、まず、命令フェッチ部 30 からオペレーションコードおよびベクトル要素数が命令パイプライン制御部 71 に入力される。

【0057】

このとき入力されるベクトル要素数は、ベクトル演算を行う要素データ数を指定するコードであり、ここでは、図 11 に示すように、3 ビットのコードである。

命令フェッチ部 30 から入力されたオペレーションコードが演算命令である場合、命令パイプライン制御部 71 が、第 1 ソースレジスタ決定回路 72 および第 2 ソースレジスタ決定回路 73 に動作開始信号を出力する。

【0058】

第 1 ソースレジスタ決定回路 72 は、命令パイプライン制御部 71 から動作開始信号が入力されると、命令フェッチ部 30 から第 1 ソースレジスタ指定コードおよびベクトル要素数を受け取る。

そして、第 1 ソースレジスタ決定回路 72 は、命令フェッチ部 30 から受け取

ったベクトル要素数に応じて、所定のレジスタを選択するための第 1 ソースレジスタ選択信号をレジスタファイル 4 0 に順次出力する。

【0 0 5 9】

すると、レジスタファイル 4 0 から、所定の第 1 ソースデータが、P R 7 6 に順次入力される。

また、第 2 ソースデータについても、第 1 ソースデータと同様の手順によって、P R 7 7 に順次入力される。

そして、演算器 7 5 が、P R 7 6, 7 7 に記憶されている第 1 ソースデータおよび第 2 ソースデータの演算を行い、演算結果をレジスタファイル 4 0 に出力する。

【0 0 6 0】

一方、命令フェッチ部 3 0 から入力されたデスティネーションレジスタ指定コードは、P R 7 8 に記憶され、同様に P R 7 9 に記憶されたベクトル要素数とタイミングを合わせられた上で、デスティネーションレジスタ決定回路 7 4 に入力される。

すると、デスティネーションレジスタ決定回路 7 4 は、演算器 7 5 が演算結果をレジスタファイル 4 0 に出力するタイミングに合わせて、所定のレジスタを選択するためのデスティネーションレジスタ選択信号をレジスタファイル 4 0 に出力する。

【0 0 6 1】

このような手順を繰り返すことにより、レジスタファイル 4 0 において、所定のデスティネーションレジスタに演算結果が順次書き込まれる。

以上のように、本実施の形態に係るベクトルプロセッサ 1 は、リングバッファをなすベクトルレジスタを備え、そのリングバッファにおける任意のアドレスを先頭アドレスとして指定可能である。

【0 0 6 2】

そのため、演算対象である複数のベクトルデータが重複する場合に、ベクトルデータそれぞれを異なるベクトルレジスタに記憶することなく、1 つのベクトルレジスタに記憶されたベクトルデータを循環的に読み込みあるいは書き込むこと

ができる。

したがって、同一のデータが重複して読み出されることを回避できると共に、必要となるレジスタリソースを減少させることが可能となり、ベクトルレジスタを用いたベクトル演算を効率的に行うことが可能となる。

【0063】

また、同一のデータが重複して読み出されることを回避できるため、消費電力が軽減される。さらに、必要となるレジスタリソースが減少することから、回路規模を縮小できると共に、処理効率を向上させることができる。

なお、本実施の形態においては、第1ソースデータおよび第2ソースデータの両方について、リングバッファをなすベクトルレジスタに記憶して演算を行うこととして説明したが、いずれか一方のみをリングバッファをなすベクトルレジスタに記憶し、他方は一般的なベクトルレジスタあるいはスカラーレジスタに記憶して演算することとしてもよい。

【0064】

また、本実施の形態においては、ベクトル要素数を命令コードによって指定することとして説明したが、レジスタファイル40あるいは他のレジスタにベクトル要素数を格納し、そのレジスタを指定することとしてもよい。

【図面の簡単な説明】

【図1】 8つの要素レジスタR0～R7を有するベクトルレジスタVRを示す図である。

【図2】 サイクル“2”におけるベクトルレジスタVRの状態を示す図である。

【図3】 サイクル“7”におけるベクトルレジスタVRの状態を示す図である。

【図4】 サイクル“8”におけるベクトルレジスタVRの状態を示す図である。

【図5】 サイクル“9”におけるベクトルレジスタVRの状態を示す図である。

【図6】 本発明を適用したベクトルプロセッサ1の構成を示す図である。

【図 7】 命令コードのデータ形式の一例を示す図である。

【図 8】 レジスタ R0～R31 それぞれに割り当てられたコードを示す図である。

【図 9】 演算ユニット 70 の内部構成を示すブロック図である。

【図 10】 第 1 ソースレジスタ決定回路 72 の構成例を示す図である。

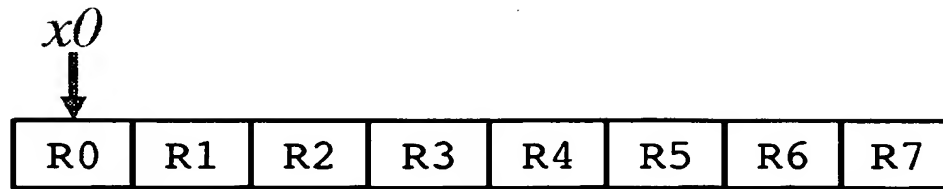
【図 11】 ベクトル要素数のコードを示す図である。

【符号の説明】

1 ベクトルプロセッサ, 10 メモリ, 20 メモリ制御部, 30 命令フェッチ部, 40 レジスタファイル, 50 ロードユニット, 60 ストアユニット, 70 演算ユニット, 71 命令パイプライン制御部, 72 ソースレジスタ決定回路, 72 a 制御部, 72 b セレクタ, 72 c インクリメンタ, 72 d カウンタ, 72 e レジスタ, 73 ソースレジスタ決定回路, 74 デスティネーションレジスタ決定回路, 75 演算器, 76～79 PR (パイプラインレジスタ)

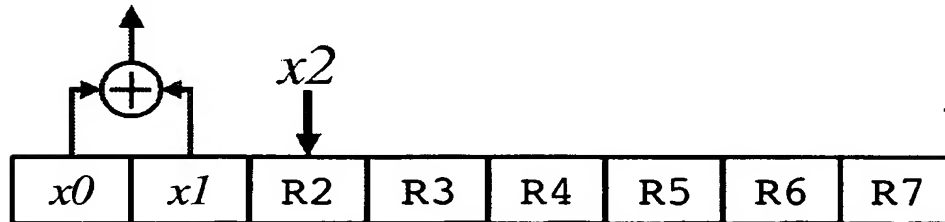
【書類名】 図面

【図 1】



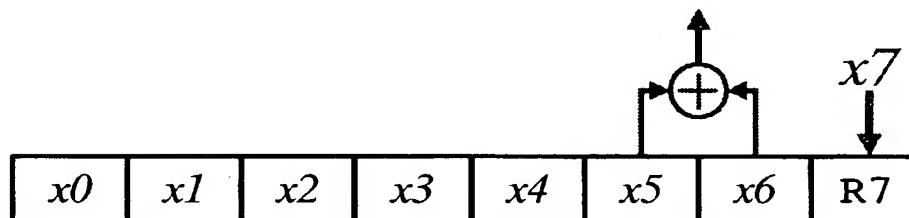
Cycle = "0"

【図 2】



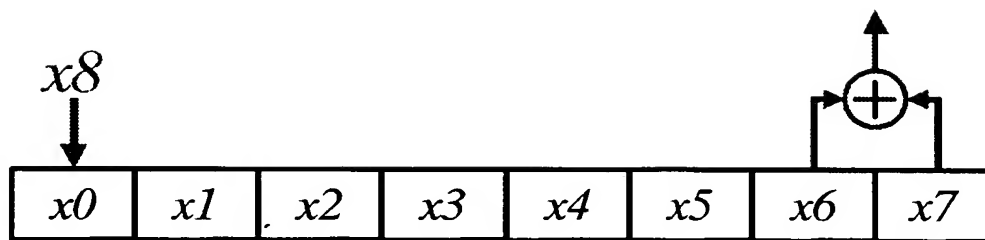
Cycle = "2"

【図 3】



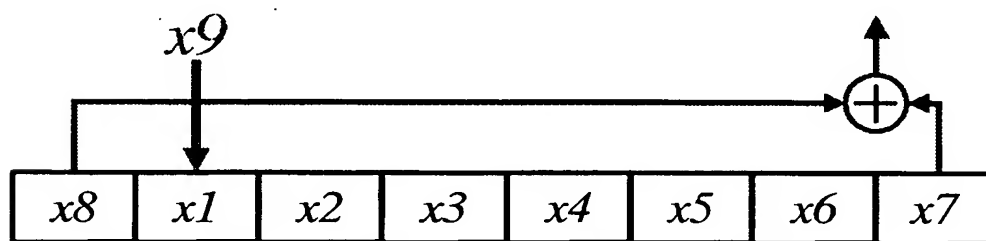
Cycle = "7"

【図 4】



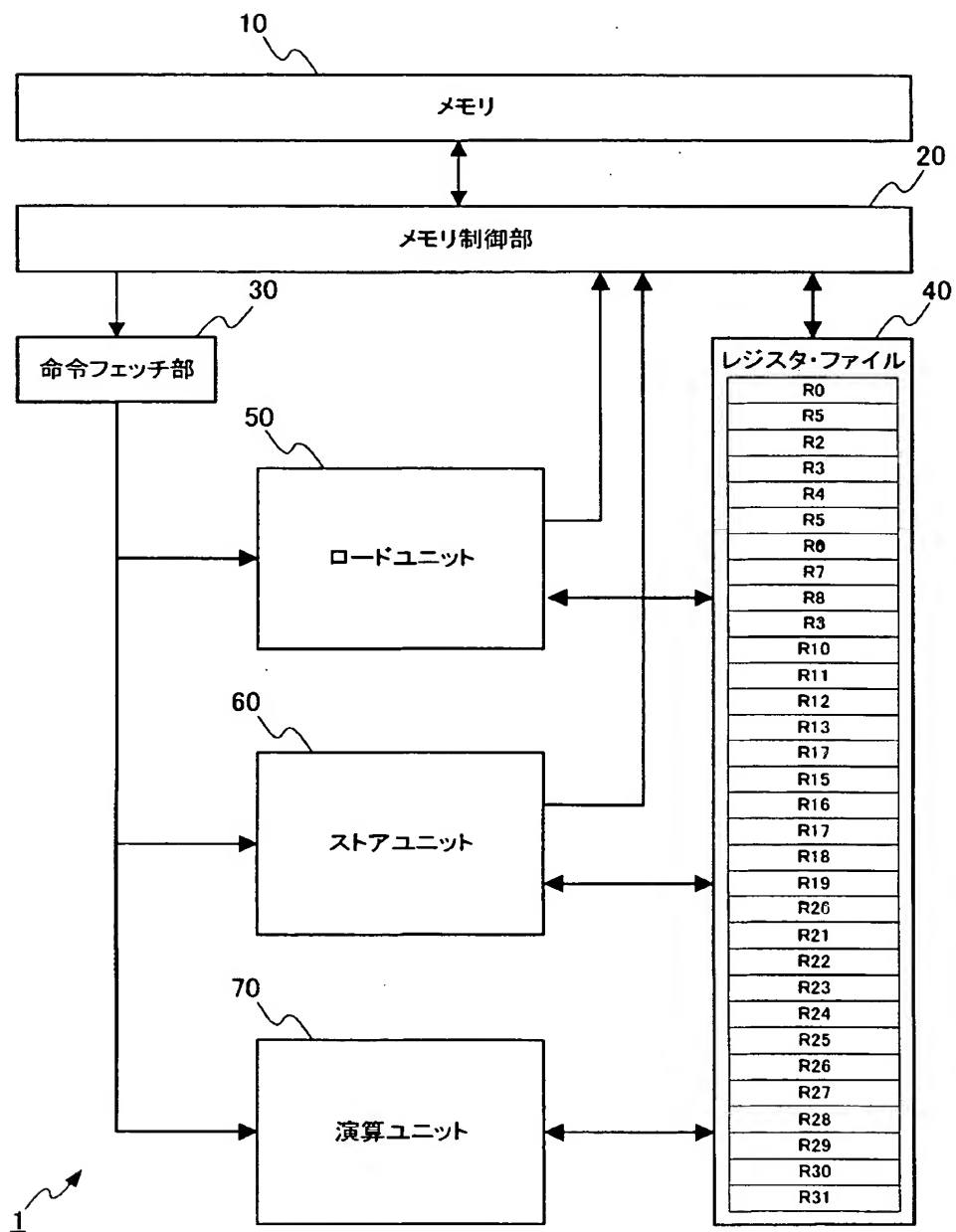
Cycle = "8"

【図 5】



Cycle = "9"

【図 6】



【図 7】

(a) ロード命令の形式

| オペレーション コード | ベクトル 要素数 | ロードレジスタ 指定コード | 基底アドレス レジスタ指定コード | アドレス修飾 レジスタ指定コード |
|----------------|-------------|------------------|---------------------|---------------------|
|----------------|-------------|------------------|---------------------|---------------------|

(b) ストア命令の形式

| オペレーション コード | ベクトル 要素数 | ストアレジスタ 指定コード | 基底アドレス レジスタ指定コード | アドレス修飾 レジスタ指定コード |
|----------------|-------------|------------------|---------------------|---------------------|
|----------------|-------------|------------------|---------------------|---------------------|

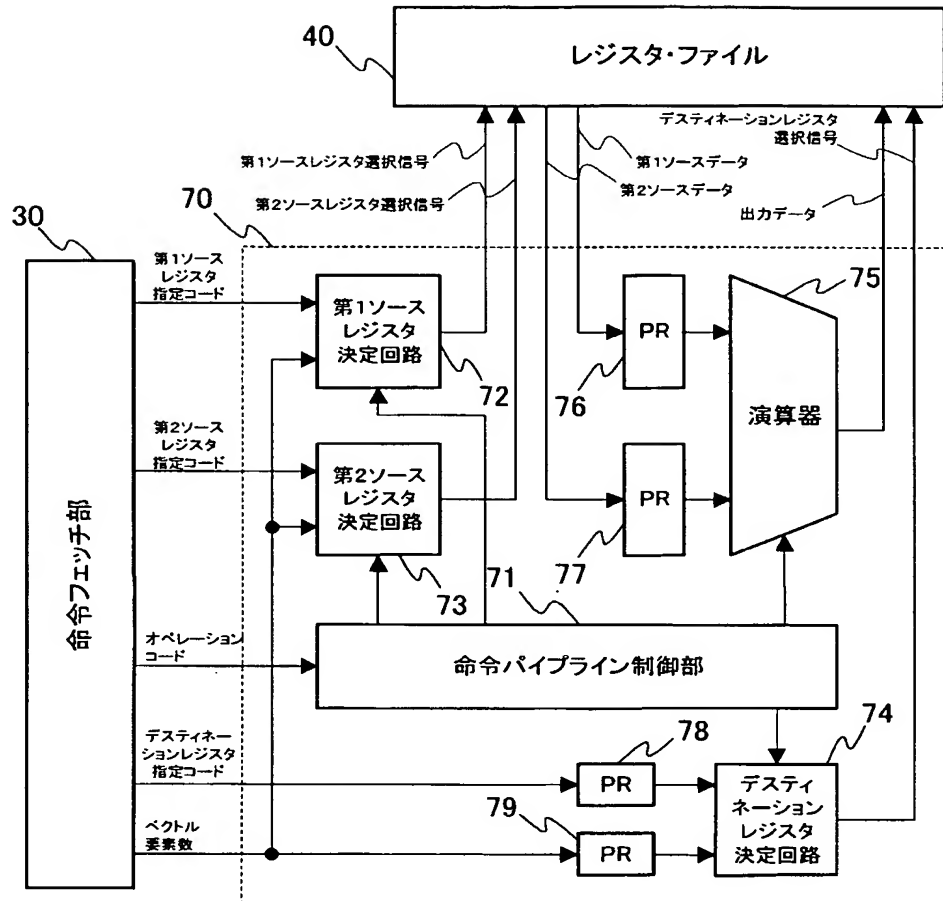
(c) 演算命令の形式

| オペレーション コード | ベクトル 要素数 | デスティネーション レジスタ指定コード | 第1ソース レジスタ指定コード | 第2ソース レジスタ指定コード |
|----------------|-------------|------------------------|--------------------|--------------------|
|----------------|-------------|------------------------|--------------------|--------------------|

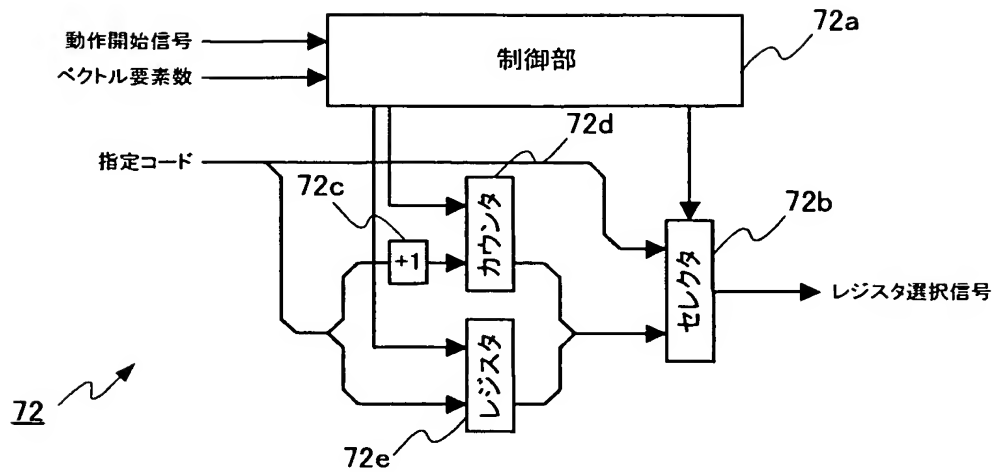
【図 8】

| レジスタ | コード | レジスタ | コード |
|------|-------|------|-------|
| R0 | 00000 | R16 | 10000 |
| R1 | 00001 | R17 | 10001 |
| R2 | 00010 | R18 | 10010 |
| R3 | 00011 | R19 | 10011 |
| R4 | 00100 | R20 | 10100 |
| R5 | 00101 | R21 | 10101 |
| R6 | 00110 | R22 | 10110 |
| R7 | 00111 | R23 | 10111 |
| R8 | 01000 | R24 | 11000 |
| R9 | 01001 | R25 | 11001 |
| R10 | 01010 | R26 | 11010 |
| R11 | 01011 | R27 | 11011 |
| R12 | 01100 | R28 | 11100 |
| R13 | 01101 | R29 | 11101 |
| R14 | 01110 | R30 | 11110 |
| R15 | 01111 | R31 | 11111 |

【図 9】



【図 10】



【図 11】

| ベクトル要素数 | 行われる演算 |
|---------|--------------------|
| 000 | 要素数 1 の演算 (スカラー演算) |
| 001 | 要素数 2 の演算 |
| 010 | 要素数 3 の演算 |
| 011 | 要素数 4 の演算 |
| 100 | 要素数 5 の演算 |
| 101 | 要素数 6 の演算 |
| 110 | 要素数 7 の演算 |
| 111 | 要素数 8 の演算 |

【書類名】 要約書

【要約】

【課題】 ベクトルレジスタを用いたベクトル演算を効率的に行うこと。

【解決手段】 リングバッファをなすベクトルレジスタを備え、そのリングバッファにおける任意のアドレスを先頭アドレスとして指定可能である。そのため、演算対象である複数のベクトルデータが重複する場合に、ベクトルデータそれぞれを異なるベクトルレジスタに記憶することなく、1つのベクトルレジスタに記憶されたベクトルデータを循環的に読み込みあるいは書き込むことができる。したがって、同一のデータが重複して読み出されることを回避できると共に、必要となるレジスタリソースを減少させることが可能となり、ベクトルレジスタを用いたベクトル演算を効率的に行うことが可能となる。

【選択図】 図5

特願 2 0 0 3 - 0 9 2 3 7 1

出 願 人 履 歴 情 報

識別番号 [0 0 0 0 0 2 3 6 9]

| | |
|----------|------------------------|
| 1. 変更年月日 | 1 9 9 0 年 8 月 2 0 日 |
| [変更理由] | 新規登録 |
| 住 所 | 東京都新宿区西新宿 2 丁目 4 番 1 号 |
| 氏 名 | セイコーエプソン株式会社 |